Abstract

Multi-GPU machines are being increasingly used in high performance computing. These machines are being used both as standalone workstations to run computations on medium to large data sizes (tens of gigabytes) and as a node in a CPU-MultiGPU cluster handling very large data sizes (hundreds of gigabytes to a few terabytes). Each GPU in such a machine has its own memory and does not share the address space either with the host CPU or other GPUs. Hence, applications utilizing multiple GPUs have to manually allocate and manage data on each GPU.

A significant body of scientific applications that utilize multi-GPU machines contain computations inside affine loop nests, i.e., loop nests that have affine bounds and affine array access functions. These include stencils, linear-algebra kernels, dynamic programming codes and data-mining applications. Data allocation, buffer management, and coherency handling are critical steps that need to be performed to run affine applications on multi-GPU machines. Existing works that propose to automate these steps have limitations and inefficiencies in terms of allocation sizes, exploiting reuse, transfer costs and scalability. An automatic multi-GPU memory manager that can overcome these limitations and enable applications to achieve scalable performance is highly desired.

One technique that has been used in certain memory management contexts in the literature is that of *bounding boxes*. The bounding box of an array, for a given tile, is the smallest hyper-rectangle that encapsulates all the array elements accessed by that tile. In this thesis, we exploit the potential of bounding boxes for memory management far beyond their current usage in the literature.

In this thesis, we propose a scalable and fully automatic data allocation and buffer

management scheme for affine loop nests on multi-GPU machines. We call it the Bounding Box based Memory Manager (BBMM). BBMM is a compiler-assisted runtime memory manager. At compile time, it uses static analysis techniques to identify a set of bounding boxes accessed by a computation tile. At runtime, it uses the bounding box set operations such as union, intersection, difference, finding subset and superset relation to compute a set of *disjoint* bounding boxes from the set of bounding boxes identified at compile time. It also exploits the architectural capability provided by GPUs to perform fast transfers of rectangular (strided) regions of memory and hence performs all data transfers in terms of bounding boxes. BBMM uses these techniques to automatically allocate, and manage data required by applications (suitably tiled and parallelized for GPUs). This allows it to (1) allocate only as much data (or close to) as is required by computations running on each GPU, (2) efficiently track buffer allocations and hence, maximize data reuse across tiles and minimize the data transfer overhead, (3) and as a result, enable applications to maximize the utilization of the combined memory on multi-GPU machines. BBMM can work with any choice of parallelizing transformations, computation placement, and scheduling schemes, whether static or dynamic. Experiments run on a system with four GPUs with various scientific programs showed that BBMM is able to reduce data allocations on each GPU by up to 75% compared to current allocation schemes, yield at least 88% of the performance of hand-optimized OpenCL codes and allows excellent weak scaling.